Department of Computer Science

CMPT 481/898 Midterm Exam

October 15, 2002

Time: 75 minutes Name: _____

Total marks: 74 Student Number: _____

Question 1 (10 marks).

A Java application has been built with event handling that implements keyboard shortcuts. In particular, the keys "CONTROL-K" mean that a particular method `kreateNode()` will be executed in the application. In the table below, fill in the blanks to show the progress of the event through all of the UI layers (there may be more blanks than needed). Write the name of the layer on the left, and what happens for the event at that layer. Write only one action per table row; low-level details are not required.

Layer	What happens
----	User presses the "CONTROL" and "K" keys on the keyboard
Device	Keyboard sends electrical signals to driver in OS
OS	The two keys are merged into one ASCII code for CTRL-K
OS	Keypress event created and forwarded to window system
Window system	Forward keypress event to toolkit of window in focus
Toolkit	Determine subscribers to keypress events
Toolkit	Call method keyPressed(KeyEvent e) of object implementing KeyListener interface for window
Application	Application code in listener keyPressed method inspects event structure for keycode and calls kreateNode()
----	Application executes the `kreateNode()` method

Question 2 (10 marks)

For each line of code in the following fragment of Tcl/Tk, state which part of a UI toolkit the line is using. Choose from the following key and write the corresponding letter beside the line (note: not all of the items in the key are valid parts of a UI toolkit). If more than one letter matches a particular line, choose the letter that best represents the line's functionality.

W = Widgets	E = Event Handling	L = Layout Management	G = Graphics
M = Window Management	D = Devices	M = Model View Controller	H = Hardware

```
1.    W    toplevel .t
2.    W    button .t.b -text "hello" -width 20
3.    W    canvas .t.c -height 400 -width 400
4.    L    pack .t.c -side top
5.    L    place .t.b -x 0 -y 0
6.    E    bind .t.c <1> "mouseClick %x %y"
7.    G    .t.c create text 200 200 -text "Hello" -tags label
```

```
8.      G     .t.c coords label 50 50
9.      W     set colour [tk_chooseColor]
10.     G     .t.c itemconfigure label -fill $colour
```

Question 3 (4 marks)

What are the four main types of constraints used in grid layout managers (as discussed in class)?

1. Where in the grid to start
2. How many cells to take up
3. Where to place the component in the cell
4. What to do when the cell resizes

Question 4 (12 marks)

Assume the following widgets exist:

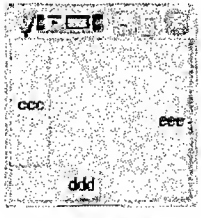
```
frame .f
button .f.a -text "aaa"
button .f.b -text "bbb"
button .c -text "ccc"
button .d -text "ddd"
button .e -text "eee"
```

The boxes below represent "." (i.e. the top window). Using your knowledge of the packer algorithm and the idea of variable intrinsic size, draw the state of the window after each of the given commands. For each, draw a box for each widget's parcel, draw the widgets inside their parcels, and clearly shade the remaining cavity in each drawing.

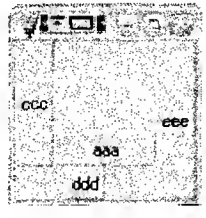
pack .e –side right –fill y

pack .d –side bottom

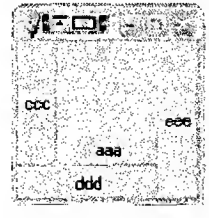
pack .c –side left –fill y



pack .f.a –side bottom –fill x



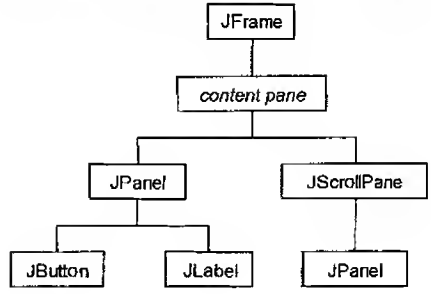
pack .f –fill both –expand true



pack configure .d –fill x

Question 5 (12 marks)

Write the Java code that would create an interface that matches the following containment hierarchy:

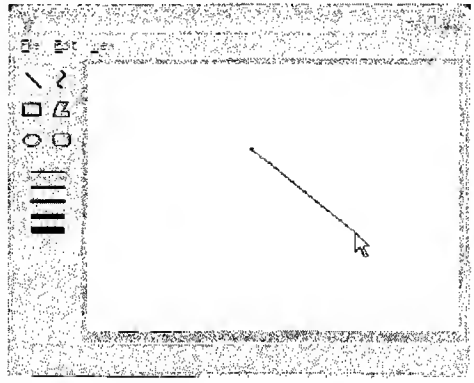



```
private void createUI () {
      JFrame frame = new JFrame();
      JPanel panel1 = new JPanel(new FlowLayout());
      JPanel panel2 = new JPanel();
      JButton button = new JButton("Button");
      JLabel label = new JLabel("Label");
      JScrollPane scrollpane = new JScrollPane(panel2);
      Container content = frame.getContentPane();
      content.setLayout(new FlowLayout());

      panel1.add(button);
      panel1.add(label);
      content.add(panel1);
      content.add(scrollpane);
    }
```

Question 6 (12 marks)

In a drawing program (such as the one pictured below), a user has selected the 'line' tool and has just created a line by releasing the mouse button after dragging. In the table below, write down the main steps that occur (in order) in each of the model, the view, and the controller, from the mouse-up event to the eventual call to `repaint()`. Write only one step per table row (there may be more rows than needed), and assume a publish-subscribe style of communication between models and views.



Model, View, or Controller	What happens
Controller	receive mouse click event
Controller	check mode = line
Controller	call model.addLine(x1,y1,x2,y2)
Model	add line to data structure
Model	notify view that a change has occurred
View	call repaint()

Question 7 (4 Marks)

What are the two main problems (as discussed in lectures) with the "erase and redraw" approach to repainting the screen?

- 'erasing' an object with the background colour can lead to creation of 'holes' in other items if the object overlaps with others
- simply redrawing the object at its new location puts it on top, which may not be the correct depth order

Question 8 (10 marks)

Write a Tcl/Tk procedure to flip a polygon horizontally on a canvas. The polygon should flip around its centre point. You may find the following useful:

- equations for scaling an object: x' = x * scaleX; y' = y * scaleY
- "<canvas> coords $item" returns a list of $item's coordinates (i.e. {x1 y1 x2 y2 ... })
- "<canvas> coords $item <list>" sets $item's coordinates to <list>
- "<canvas> bbox $item" returns a list of the bounding box for $item

```
proc flipHorizontal {c id} {

    # find item's centre x
    set box [$c bbox $id]
    set cx [expr ([lindex $box 2] + [lindex $box 0]) / 2.0]

    # translate to x-origin
    $c move $id -$cx 0

    # flip each x point in coordinate list for id
    set newCoords {}
    set coords [$c coords $id]
    for {set i 1} {$i < [llength $coords]} {incr i 2} {
       set x [lindex $coords [expr $i - 1]]
       set y [lindex $coords $i]

       # scaling by a factor of -1 flips the point
       set newX [expr $x  * -1]
       lappend newCoords $newX $y
    }
    # change to coordinates of the item
    $c coords $id $newCoords

    # translate back to original position
    $c move $id $cx 0
}
```